# Machine Learning in the Presence of an Adversary: Attacking and Defending the SpamBayes Spam Filter

*Udam Saini*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-62
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-62.html

May 20, 2008

| Report Documentation Page | *Form Approved* *OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| 1. REPORT DATE **20 MAY 2008** | 2. REPORT TYPE | 3. DATES COVERED **00-00-2008 to 00-00-2008** |
| 4. TITLE AND SUBTITLE **Machine Learning in the Presence of an Adversary: Attacking and Defending the SpamBayes Spam Filter** | | 5a. CONTRACT NUMBER |
| | | 5b. GRANT NUMBER |
| | | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | | 5d. PROJECT NUMBER |
| | | 5e. TASK NUMBER |
| | | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Electrical Engineering and Computer Sciences,Berkeley,CA,94720-1700** | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSOR/MONITOR’S ACRONYM(S) |
| | | 11. SPONSOR/MONITOR’S REPORT NUMBER(S) |
| 12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited** | | |
| 13. SUPPLEMENTARY NOTES | | |
| 14. ABSTRACT **see report** | | |
| 15. SUBJECT TERMS | | |

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **54** | |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std Z39-18

# Machine Learning in the Presence of an Adversary: Attacking and Defending the SpamBayes Spam Filter

by Udam Saini

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

**Committee:**

Professor Anthony D. Joseph
Research Advisor

(Date)

* * * * * * *

Professor Doug Tygar
Second Reader

(Date)

**Machine Learning in the Presence of an Adversary: Attacking and Defending the SpamBayes Spam Filter**

## Abstract

Machine Learning in the Presence of an Adversary: Attacking and Defending the SpamBayes Spam Filter

by

Udam Saini
Master of Science in Computer Science

University of California at Berkeley

Professor Anthony D. Joseph, Research Advisor

Machine learning techniques are often used for decision making in security critical applications such as intrusion detection and spam filtering. However, much of the security analysis surrounding learning algorithms is theoretical. This thesis provides a practical evaluation of the algorithms used by SpamBayes, a statistical spam filter, to determine its ability to correctly distinguish spam email for normal email when learning in the presence of an adversary.

This thesis presents both attacks against SpamBayes and defenses against these attacks. The attacks are able to subvert the spam filter by both causing a high percentage of false positives and false negatives. With only a 100 attack emails, out of an initial training corpus of 10,000, the spam filter's performance is sufficiently degraded to either cause a denial of service attack or successfully allow spam emails to bypass the filter. The defenses shown in this thesis are able to work against the attacks developed against SpamBayes and are sufficiently generic to be easily extended into other statistical machine learning algorithms.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Statistical machine learning algorithms are used in many areas of computing in order to make intelligent decisions on the classification of data. These algorithms learn on a set of data, either in an online or offline manner, to produce a set of rules and patterns it can base future data classifications.

These algorithms are used to adapt the application to the changing flow of data around it by constantly training or retraining on new information. Since the algorithm is constantly adapting itself to new information, the application does not have to be changed or rewritten to accommodate a shifting computing environment. This property also makes the application more malleable to its use.

Often, security sensitive tasks, such as network intrusion/extrusion detection, financial fraud detection, virus and worm detection, and spam filtering use statistical machine learning algorithms to improve data classification [5, 12, 13, 17, 18, 23]. The key strength of machine learning is its adaptability; however, this strength becomes a weakness when an adversary manipulates the learner's environment. With the continual growth of malicious activity and electronic crime, the increasingly widespread adoption of learning makes assessing the vulnerability of machine learning systems an essential security task.

In this thesis, I look at SpamBayes[1] (spambayes.sourceforge.net) [22], a statistical spam filter that uses a naive Bayes learning algorithm. Many other spam filters, such as SpamAssassin (spamassassin.apache.org) and BogoFilter (bogofilter.sourceforge.net), use a similar Bayesian classifier. The primary difference amongst these learners is in the email tokenization process.

[1]a popular spam filter with over 700,000 downloads

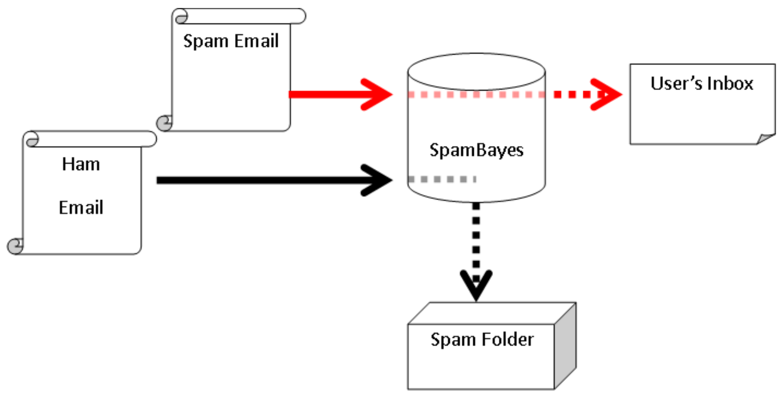

Figure 1: Attacks discussed in this thesis have two main goals. Either the attacker attempts to cause SpamBayes to misclassify ham emails as *spam* or causes SpamBayes to misclassify spam emails as *ham* emails.

This thesis focuses on SpamBayes as it uses a pure machine learning algorithm for spam filtering, and it has received prior attention in the academic community [16]. Since other spam filters use a similar method for classifying spam, the attacks and defenses presented may readily apply to other spam filters as well. It is also important to note that other spam filters, such as SpamAssassin, may only use the learner as part of its filtering system.

I demonstrate the effectiveness of subverting the learning process of spam filters by presenting two attacks against SpamBayes. A key difference, in the method of attack, is that the adversary sends malicious emails that alter the trained model of the spam filter rather than simply attempting to bypass the filter. As illustrated in Figure 1, the adversary can either send attack email such that future legitimate email (or *ham*) becomes classified incorrectly or future spam email from the adversary is placed into the user's inbox.

In the first attack, the adversary can either target a specific important email to be misclassified or all legitimate email. In the first variation of the attack, the adversary may not want a competitor's email to be sent to their customer. In the latter variation, the adversary may be a spammer who wishes to frustrate the user until the user turns off his spam filter due to many normal legitimate email being filtered into the spam folder. Once the spam filter is turned off, all spam email will be seen in the user's inbox. In the second attack, the adversary sends email that appears to be legitimate to the user, and thus is trained as a non-spam email, but contains important header information that is specific to the adversary. After the training phase, the adversary can later send spam email that contains this header information, so that the spam email is misclassified.

Two different and orthogonal defenses are presented to prevent email classification error due to adversarial attacks on the training process. Although these defenses are preliminary in nature, initial experimental evidence shows that the defenses are successful in preventing some of these attacks from succeeding. The *Reject On Negative Impact (RONI) defense* tests the impact each email has on the trained model, and if this impact is largely negative, the email is removed from the training set. Orthogonal to this defense is the *dynamic threshold defense*, which dynamically alters SpamBayes' classification thresholds based on the shifting data rather than maintaining fixed thresholds.

My experimental results indicate that both of the attacks presented are extremely effective in misclassifying emails. These attacks, and their variations, work well even when only 1% of the training data is poisoned with the adversary's attack emails. The defenses are currently only effective in preventing the success of these attack emails when the attacks affect features present in current emails. For instance, these defenses are not very effective when the malicious emails only influence features present in future emails the spam filter has not yet seen.

In the chapters that follow, I first discuss other related work regarding learning in the presence of an adversary (Chapter 2). Chapter 3 provides a discussion on the algorithms used in SpamBayes and the training model assumed throughout the thesis. Chapter 4 presents the framework used for the attacks along with a discussion of the attacker's capabilities in carrying these attacks through against the spam filter. Chapter 5 motivates and explains two orthogonal defense strategies. Next, Chapter 6 illustrates the success of the attacks and defenses, followed by a discussion of these results and future research directions (Chapter 7).

# Chapter 2

# Related Work

There are several papers that analyze the security of different machine learning techniques. This section presents several related pieces of work in this area. Many works in the literature focus on exploiting the application after the learning phase. Fogla and Lee are able to fool intrusion detection systems through the use of a *polymorphic blending attack* [7], an attack that encrypts malicious traffic so as to appear identical to normal traffic. Wittel and Wu along with Lowd and Meek demonstrate a *good words attack* that adds words that are indicative of non-spam to spam emails [15, 24]. This makes spam email appear closer to non-spam email, especially if a large number of good words are added to the email to counteract the spam words.

In a more theoretical perspective, Lowd and Meek look at spam classifiers and try to reverse engineer the classification system [14]. They try to determine the adversary's cost of the attacks and try to seek the minimal cost message that bypasses the spam filter. Kearns and Li bound the classification error the adversary can cause, in the *probably approximately correct* learning framework, with control over a fraction of the training set [10].

Newsome, Karp, and Song explore attacks against the learning component of polymorphic virus detectors [19]. These attacks involve adding extra unnecessary information to the malicious data, a *red herring attack*. They focus primarily on conjunction learners, exploiting specific weaknesses against this type of learner. Also, suggested in the paper, is the *correlated outlier attack*, which causes a naive-Bayes-like learner to train on additional spurious features of an attack instance as being malicious. This will lead to the filter blocking innocuous traffic that have traces of the spurious features causing a Denial of Service.

Chung and Mok present another Denial of Service attack against the Autograph worm signature generation system [2]. Autograph works by detecting infected nodes by behavioral scanning patterns, and then infers additional behavioral rules by monitoring the traffic patterns of the infected nodes. In the attack, a node convinces Autograph it is malicious by scanning the entire network, and then later sends traffic that mimics other legitimate traffic, which will effectively cause normal traffic to be blocked.

# Chapter 3

# Spambayes

SpamBayes classifies using token scores based on a simple model of spam status proposed by Robinson [20], based on ideas by Graham [9], together with Fisher's method for combining independent significance tests [6]. This method is similarly used by other statistical spam filters such as the ones found in SpamAssassin and BogoFilter.

SpamBayes tokenizes the header and body of each email before constructing token spam scores. Robinson then assumes that the presence or absence of possible tokens independently affects an email's spam status. For each possible token $w$, the raw token spam score

$$\mathrm{P}_S(w) \quad = \quad \frac{N_H N_S(w)}{N_H N_S(w) + N_S N_H(w)} \tag{3.1}$$

is computed from the training set counts $N_S$, $N_H$, $N_S(w)$, $N_H(w)$—the number of *spam* emails, *ham* emails, *spam* emails including $w$ and *ham* emails including $w$.

Robinson smooths $\mathrm{P}_S(w)$ through a convex combination with a prior belief $x$, weighting the quantities by $N(W)$ (the number of training emails with $w$) and $s$ (chosen for strength of prior), respectively:

$$f(w) \quad = \quad \frac{s}{s+N(w)}x + \frac{N(w)}{s+N(w)}\mathrm{P}_S(w) \ . \tag{3.2}$$

Robinson combines the smoothed token scores with two applications of Fisher's method into a message score

$$I \quad = \quad \frac{1+H-S}{2} \ \in \ [0,1] \ , \tag{3.3}$$

$$\text{where} \quad H \quad = \quad 1 - \chi^2_{2n}\left(-2\sum_{w\in E'}\log f(w)\right) \tag{3.4}$$

and $\chi^2_k(\cdot)$ denotes the cumulative distribution function of the chi-square with $k$ degrees of freedom. $S$ is defined like $H$, with $f(w)$ replaced by $1-f(w)$. SpamBayes sums up to

150 of the tokens having scores furthest from 0.5 that are outside the interval $[0.4, 0.6]$ (the set $E$' in (3.4)). SpamBayes predicts by thresholding against two user-tunable thresholds $\theta_0, \theta_1$, with defaults $\theta_0 = 0.15$ and $\theta_1 = 0.9$:

$$\hat{y}_{sb}(E) \quad = \quad \begin{cases} \textsc{Spam} \ , & I > \theta_1 \\ \textsc{Unsure} \ , & \theta_1 \geq I > \theta_0 \\ \textsc{Ham} \ , & \text{otherwise} \end{cases} \ .$$

The inclusion of an *unsure* category prevents us from purely using misclassification rates (false positives and false negatives) for evaluation. I must also consider *spam-as-unsure* and *ham-as-unsure* emails. As described below, I treat *unsure* classifications as errors on ham emails, though not as serious as *spam* classifications.

## 3.1 Email Training

In this section, I describe the typical SpamBayes usage scenarios that I assume in designing my attacks. SpamBayes automatically produces a classifier from labeled examples that it then uses to label future emails. An email can be given the labels *spam* (bad, unsolicited email), *ham* (good, legitimate email), or *unsure* (SpamBayes is not confident one way or the other). The classifier is constructed from a labeled *training set* of emails of ham and spam.

Email clients use these labels in different ways—some clients filter email labeled as *spam* and *unsure* into "Spam-High" and "Spam-Low" folders, respectively, while other clients only filter email labeled as *spam* into a separate folder. Since the typical user reads most or all email in their inbox and rarely (if ever) looks at the spam/spam-high folder, the *unsure* labels can be problematic. If *unsure* messages are filtered into a separate folder, users must periodically read the messages in that folder to avoid missing important emails. If instead *unsure* messages are not filtered, then the user faces those messages when checking the email in their inbox. Too much *unsure* email is almost as troublesome as too many false positives (ham labeled as *spam*) or false negatives (spam labeled as *ham*). In the extreme, if everything is labeled *unsure* then the user obtains no time savings at all from the filter.

It can be considered that there is some probability $p$ that the user checks an email in their "Spam-Low" folder, which is greater than the probability a user looks at their "Spam-High" folder, but less than the probability a user checks the emails in their inbox. Thus, with probability *1-p* the user does not see a ham email that gets classified as *unsure*, and with probability $p$ the user does see a spam email that is classified as *unsure*. Thus, an *unsure* classification cannot be considered a success for SpamBayes for either ham emails or spam emails, but neither of these are as detrimental as pure false positives and false negatives since the "Spam-Low" folder is looked at only infrequently.

In my scenarios, an organization uses SpamBayes to filter multiple users' incoming email[1] and trains on everyone's received email. SpamBayes may also be used as a personal email filter, in which case the presented attacks and defenses are likely to be equally effective.

To keep up with changing trends in the statistical characteristics of both legitimate and spam email, I assume that the organization retrains SpamBayes periodically (e.g., weekly). While the training set could be constructed in several different ways, for simplicity, we assume a *self training* procedure where the filter is retrained on all received ham and spam email, which has been labeled by the previous version of the classifier (potentially with some human-provided corrections of the classifier's mistakes). Other practical methods for creating the training set include: manually inspecting and hand labeling a randomly selected fraction of email; retraining only on the mistakes made by the previous version of the classifier, using correct labels provided by a human.

[1]I use the terms *user* and *victim* interchangeably to refer to either an organization or individual member; the meaning will be clear from context.

# Chapter 4

# Attacks

## 4.1 Framework

To better understand potential attacks, I first analyze the *security goals* of the spam filter, and the *threat model*. The security within a system is established to prevent a security goal from being violated while a threat model describes the adversary in detail, explaining the adversary's capabilities. For spam filtering, there are usually multiple security goals.

In this section, I analyze the security goals and threat model for SpamBayes in the context of a general taxonomy for learning systems developed by Barreno et al [1]. A spam filter's goal is to prevent most, if not all, spam email from reaching the user's inbox. Another goal is to keep all legitimate email from being filtered away into the spam folder. If either one of these goals are violated, the security of the system can be said to be compromised.

There is a clear connection between false negatives and violation of the integrity goal: spam email reaches the user's inbox. Likewise, false positives tend to violate the availability goal because the learner itself denies legitimate email by placing it in the spam folder.

In my threat model, the attacker has knowledge of the algorithms used by SpamBayes. The adversary, in some cases, may have knowledge of some of the future emails that will be received by the spam filter (e.g. specific words that will appear eventually) after the learner has finished its training. Another important attacker capability is the ability to generate email that will be used in the training set when the learner retrains.

I limit the attacker from having arbitrary control over the training set. If the adversary has total control of the data learned by the system, it would be difficult for the system to learn anything useful, and would be nearly impossible to defend against. Thus, as a reasonable limit for my attacks, I prevent the adversary from having more than 10% control of the training data.

Barreno et al developed a taxonomy with three axes that categorizes attacks against learning systems:

***Influence***

- *Causative attacks* affect the learning phase by sending malicious email, which ends up being used in the training set.
- *Exploratory attacks* exploit errors in email classifications, but do not affect training.

***Security violation***

- *Integrity attacks* compromise the system via false negatives.
- *Availability attacks* cause denial of service, usually via false positives.

***Specificity***

- *Targeted attacks* focus on a particular email.
- *Indiscriminate attacks* encompass a wide range of emails.

The first axis describes the capabilities of the attacker: whether (a) the attacker has the ability to influence the training data that is used by the spam filter (a *causative attack*) or (b) the attacker does not influence the learned model, but can send new instances to the classifier and possibly observe its decisions on these carefully crafted instances (an *exploratory attack*). In one sense, *causative attacks* are more fundamentally learning attacks than *exploratory attacks* are: while many types of systems perform poorly on cleverly modified instances, only systems that learn from data can be led by an attacker to form incorrect models or choose poor hypotheses. On the other hand, the hypotheses produced by learning algorithms have certain regularities and structures that an attacker may be able to exploit in an *exploratory attack*, so it is certainly worthwhile to consider them carefully alongside *causative attacks*.

The second axis indicates the type of security violation the attacker causes: (a) to create false negatives, in which spam email slip through the filter (an *integrity* *violation*); or (b) to create a denial of service, usually by inducing false positives, in which ham email are incorrectly filtered (an availability violation).

The third axis refers to how specific the attacker's intention is: whether (a) the attack is highly *targeted* to degrade the classifier's performance on one particular email or (b) the attack aims to cause the classifier to fail in an *indiscriminate* fashion on a broad class of messages. Each axis, and especially this one, is actually a spectrum of choices.

In this thesis, I discuss attacks that would fall under the *causative availability* and *causative integrity attacks.*

- *Causative Availability attacks* attack the learning component of the system and attempt to cause innocuous data to be flagged as malicious and filtered away.
- *Causative Integrity attacks* attack the learning component of the system and attempt to cause malicious data to be deemed innocent and let through.

For spam messages, *causative availability attacks* manipulate the spam filter's training data to increase the number of false positive, that is the number of ham emails incorrectly classified as *spam.* The *causative integrity attacks* increase the number of false negatives, that is the number of spam emails incorrectly classified as *ham* through training data manipulation.

The attacks discussed in this thesis fall under both *indiscriminate* and *targeted attacks.* Specifically, one of the *causative availability attacks* presented is a *targeted attack* as the adversary targets a specific ham message to be classified as *spam.* In a variation of this attack, the adversary targets all ham messages to be misclassified, which can be considered an *indiscriminate attack.* In another *indiscriminate attack*, the adversary causes all spam emails sent by the adversary to be classified as *ham.* This attack is not completely indiscriminate as the adversary does not attempt to cause the misclassification of spam emails that are sent by other spammers.

## 4.2 Attacks Overview

An adversary who is trying to cause their to be spam seen, has two primary long-term goals when trying to subvert the spam filter: (1) to get spam messages successfully through the spam filter or (2) to cause enough ham messages to be misclassified such that the user turns off the spam filter. Thus, it is the job of the spam filter to successfully classify both ham and spam messages properly when being bombarded by

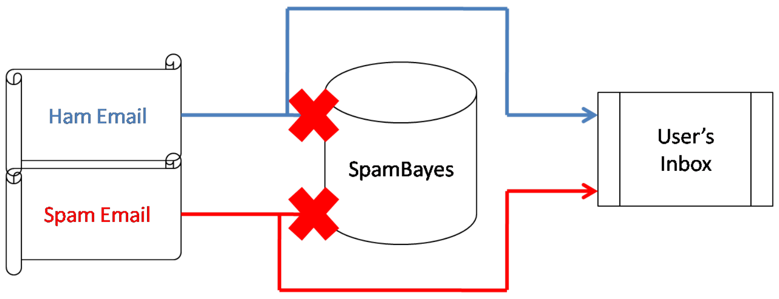

Figure 2: Once the user grows frustrated enough with the inaccuracy of the spamfilter, the user will turn it off causing all email to reach the user's inbox.

attack emails. Otherwise the adversary can take advantage of either type of email misclassification. Both false positives and false negatives must be minimized.

Much of the previous literature in this area focuses on adversarial attacks with short-term goals that try to send spam successfully through the spam filter. These attacks are against a trained filter, and do not attempt to poison the learning process. At best, the adversary will be able to successfully cause all spam email to be misclassified. Usually, however, only a subset of spams will bypass the filter.

An attack that causes availability problems for the user by misclassifying legitimate emails can potentially be more dangerous than the style of attacks discussed above. These attacks will eventually force the user to turn off the spam filter, as shown in Figure 2, which completely eliminates any effectiveness the spam filter may have had. Any and all spam will be seen by the user. It is likely that spammers will turn more of their resources towards this goal, if they haven't already, as this type of attack causes more spam messages (all of them) to reach the user's inbox.

Attacks discussed in this thesis have a delayed effect as the adversary has more of a long term goal. This is due to the causative nature of these attacks: that is, these attacks are attacking the learning component of the filter. These attacks attempt to interfere with the training data, which causes the filter to learn on malicious information. Only after the attack emails have been trained on by the filter, which may take some time depending on the method for emails to enter the training set, can the attack become effective.

I assume that the attacker can send emails that the victim will use for training, but place the following significant restriction: attackers may specify arbitrary email bodies but not any arbitrary component of the header. Certain header fields cannot be controlled by the attacker, and I prevent the adversary from manipulating particular

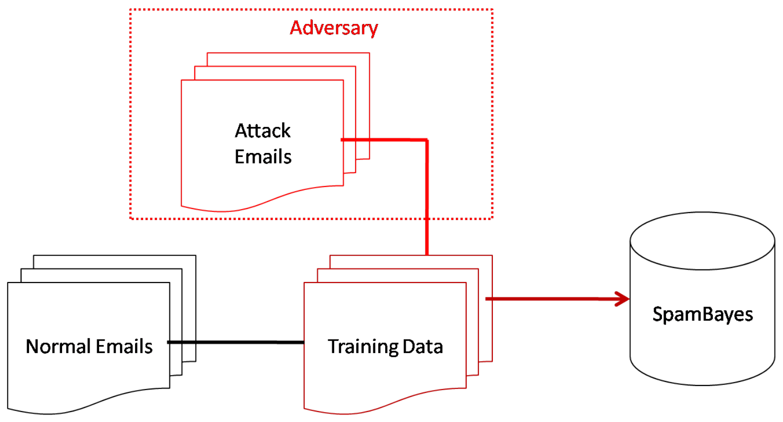

Figure 3: The adversary sends email to the user, which gets collected along with normal email, and is used as training data. This trained data is then learned on by SpamBayes.

header fields. Given this restriction, I examine the effect of allowing the attacker to control an arbitrarily large portion of the training data—an assumption that is realistic for many uses of SpamBayes. In my experiments, however, I only model instances where the attacker may have control of up to ten percent of the training set.

An attacker normally is able to send mail that is only labeled as *spam* by the user. However, it is possible for attackers to send mail that is likely to be labeled as a *ham* if the content of the message is similar to other legitimate messages a user might receive in his or her inbox. For instance, an attacker could send mail with text from a newspaper article or an RSS feed, which may look very similar to other messages the user would receive. It is also possible for these cleverly crafted emails to be mislabeled by the spam filter if the filter is used to label raw data since much of the text would be similar to normal English text found in standard email.

Two very different attacks are discussed in this thesis: the *focused attack* and the *pseudospam attack*. These attacks are both causative attacks, but the *focused attack* attempts to subvert the availability of a particular email message while the *pseudospam attack* violates the filter's integrity by causing the filter to allow the adversary's spam messages through into the inbox. The *focused attack* can be considered a *targeted* attack while the *pseudospam attack* falls in between the spectrum of *targeted* and *indiscriminate*.

In the *focused attack*, the adversary is only targeting a single specific ham email of particular importance to the adversary (and likely also to the user). The adversary's goal is to prevent the user from seeing this message by causing the spam filter to classify this particular message as *spam*. With this scenario, the adversary does not care how any of the user's other messages may be classified and would not mind if the spam filter actually becomes more effective overall as long as this message is misclassified.

In the *pseudospam attack*, the attacker wants the spam filter to misclassify his spam emails so that the spams are visible to the user. Ham messages as well as other spam messages may still be correctly classified by the spam-filter and the attack can still be successful as long as the adversary's spam emails are classified as *ham.* A wide range of spam emails can still be sent by the attacker, but certain information that the adversary controls will cause his spam emails, regardless of the content type, to reach the user's inbox.

It is interesting to note that the adversary's goals are not completely against the end goals of the user. This is due to the adversary having specific goals that only target a few emails instead of a larger set of the emails the user receives. The user wants all emails to be classified properly, but the adversary only needs a small subset of these to be misclassified to be successful due to the *targeted* nature of these attacks.

## 4.3 Focused Attack Details

My first attack, the *focused attack*, assumes knowledge of a specific legitimate email the attacker wants to be incorrectly blocked by the victim's spam filter. This is a *causative targeted availability* attack. In the *focused attack*, the attacker sends attack email to the victim that contain words likely to occur in the target email.

For example, consider a malicious contractor wishing to prevent the victim from receiving email messages with competing bids. The attacker sends spam emails to the victim with words such as "bids," "contract," "terms," and "deadline" along with some more specific words such as the name of the competing company. The bid message may even follow a common template of the competitor, making the attack easier. Figure 4 illustrates an example where the adversary inserts words into an email that may be used in a future email from Google to Yahoo.

The idea is to send attack emails that contain many words likely to occur in the targeted email. When the victim trains SpamBayes with these attack emails marked as *spam*, the words in the attack emails will have higher spam score. The future legitimate email is more likely to be marked as *spam*, because it is likely to contain spurious words from the attack email. However, the attacker needs to be careful in guessing the correct words as having a large proportion of words in the attack emails that are not in the actual targeted email will cause the email to have a lower spam score. This is due to

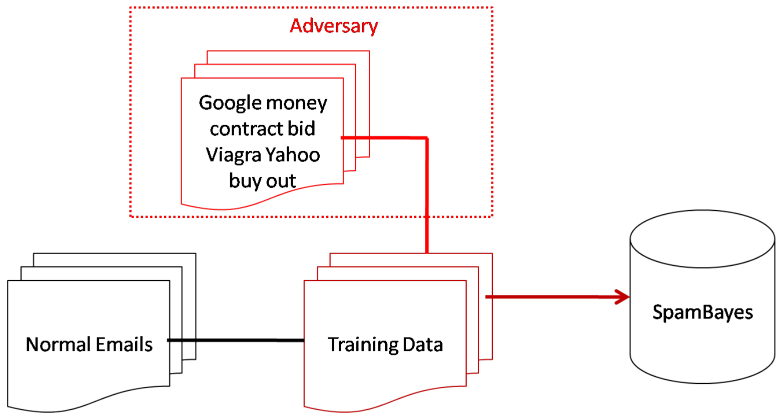

Figure 4: In the *focused attack*, the adversary sends email containing words that are likely to appear in an email from Google, who wishes to buy-out Yahoo. In this example, the words are jumbled and spam words, such as Viagra, are added to make it appear more like spam.

the fact, that words not included in the attack emails, which end up in the targeted email, become more indicative of *ham* in the SpamBayes learning model.

The attacker may have varying levels of knowledge about the targeted email. In the extreme case, the attacker might know the exact content of the targeted email, and so include all of its words. More realistically, the attacker only guesses a fraction of the email's content. In this case, the attack email might also include additional words drawn from a general distribution over email text.

If the attacker does not have any knowledge about the target email, the adversary can instead send emails, that when trained by SpamBayes, will cause any legitimate email to be classified as *spam*. In the optimal case, the adversary can simply send every possible token in the attack emails. An approximation to this technique is to simply send an entire English dictionary or the entire set of unique words taken from public newsgroups, such as USENET. The latter method would be a bit more effective as this would contain common misspellings or slang not present in dictionaries.

This variation of the attack can also be used by spammers. The spammer can send these dictionary emails, which will cause ham emails to be filtered into the spam folder. With enough misclassifications (say, ten percent or more), the user may grow frustrated and turn off the spam filter. Once the spam filter is turned off, any spammer can freely send email with any spam content, and the user will see it in his inbox.

To generate the attack emails, I create the contents of the attack email by randomly guessing a fraction $p$ in .1, .3, .5, .9, and 1.0 of the words in the targeted email.

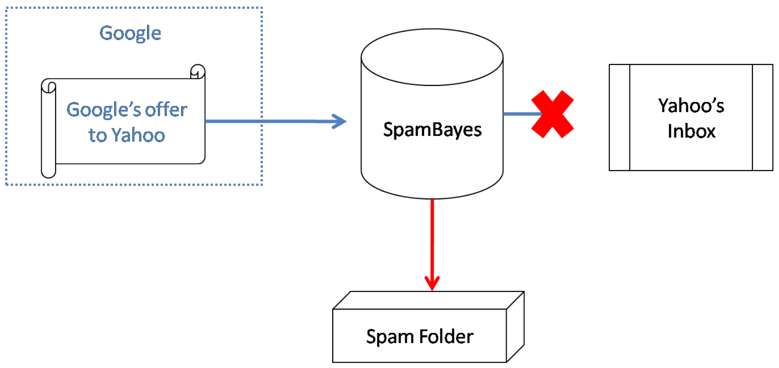

Figure 5: After the attack emails are processed by SpamBayes, the actual email from Google becomes classified as spam, which is sent to the spam folder instead of reaching Yahoo's inbox.

To ensure that a fair selection of words are successfully chosen by the attacker, an exact fraction of the words are selected in the attack emails. The words guessed incorrectly by the adversary are drawn from a random selection of words from a USENET corpus [21]. The header contents of these emails are drawn from from normal spam headers that would be found in spam messages in order to correctly simulate that the email is coming from a malicious adversary.

For the more *indiscriminate* version of the attack, I send entire dictionaries based off of the English dictionary [8] or USENET postings in each of the attack emails. The generated dictionary emails contain spam headers similar to other spam messages to simulate the email being sent from a spammer. These emails are repeatedly sent by the adversary in order to cause all tokens present in the emails to be high indicators of *spam*.

Some care is taken to creating these headers and combining this with the body contents of the email. The content-type and other Multipurpose Internet Mail Extensions (MIME) headers from the spam messages are modified or removed so that the main content of the email will be parsed correctly by the spam filter. SpamBayes looks carefully at these MIME header types, and uses these to help determine the message format. If these headers are not maintained correctly, it is likely that the body contents of the email will be ignored or likely parsed incorrectly causing SpamBayes to learn unintended information. The corpus used for ham and spam email have the dates and several other header fields modified so that there are no obvious artifacts that can be used to easily distinguish between ham and spam email.

The modified header fields are also values that the attacker can modify before

| **Header Field** | **Modified?** | **Description** |
|---|---|---|
| **Received** | No | Generated by domains, and cannot be changed by the adversary |
| **To** | No | Specifies the recipient of the email message |
| **Content-Type** | Yes | Changes the interpretation of the message body by SpamBayes |
| **Content-Transfer-Encoding** | yes | Changes the decoding of the message body |

Table 1: The modified column specifies whether a standard spam header from the corpus was modified. Only header fields that an adversary can change, and would have an affect on the interpretation of the message body were modified. Other MIME headers were also modified, such as the other Content-* fields, but SpamBayes only looks at Content-Type and Content-Transfer-Encoding fields to determine how to parse the body.

sending the email to the user. Thus, I am not giving an unfair advantage to the attacker, and in doing so, I ensure the emails are parsed properly.

## 4.4 Pseudospam Details

In contrast to the previous attack, the *pseudospam attack* attempts to misclassify spam emails that the attacker sends to the user. To achieve this, the attacker sends mail that gets trained as *ham* instead of as *spam*.

To cause a human to misclassify a training email as *ham*, the adversary sends messages that look similar to normal email, such as newspaper articles, journals, or other ham emails drawn from the adversary's own inbox. However, some of the header information contained in these emails will be similar to future spam emails the adversary sends. Not all header tokens will be the same, but there will be enough similarities that a large portion of the tokens in the headers become high indicators of *ham*.

With enough of these pseudospam emails added to the *ham* portion of the training set, the probabilities of the words contained in the spam header become much more likely to indicate *ham* according to the SpamBayes classifier. Since the spam message is not affected, the adversary can successfully send different types of spam messages in the body of an email to the user. The only necessary constant is that the spam headers are relatively similar to the headers present in the earlier attack emails. Figure 6 demonstrates the adversary's goal with the *pseudospam attack*.

To generate an attack email, I first randomly select a single ham and spam email from the corpus. Next, I combine the ham email with the header from the spam

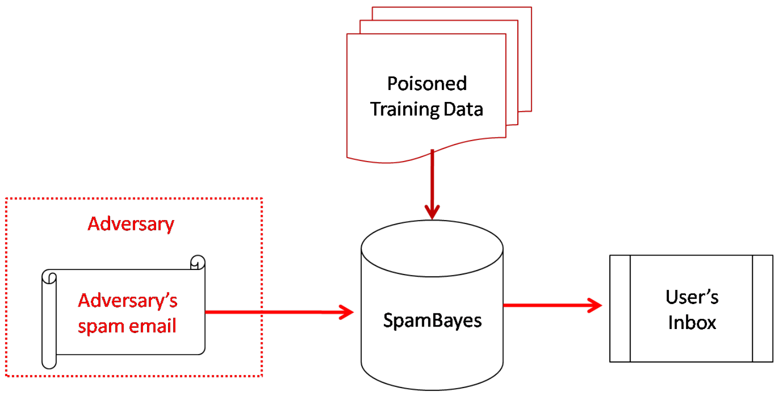

Figure 6: After training on the malicious emails in the *pseudospam attack*, the adversary will be able to send any spam email to the user, which will pass through the filter and successfully reach the user's inbox.

email. This is done so that the attack email has contents that are similar to other legitimate email messages. The header fields that may modify the interpretation of the body, such as those described in the *focused attack* above, are taken from the ham email instead. This is still within limits of the adversary's capabilities as all of these headers are all generated before the email is sent to the user over the Internet.

The spam email chosen in each of the attack emails is the same, so that the header fields are nearly the same except for the header changes described above. This is done to give more *ham* weight to the header tokens the adversary will eventually use in the spam emails that are sent to the user's inbox.

To prevent the attacker from having an unfair advantage, I limit the header fields that the adversary has control over. Received header fields for instance are not controlled by the attacker, so these are never kept the same between different emails. Instead, these header fields (along with others that an adversary cannot control under normal circumstances) are drawn randomly from the set of spam emails in the corpus.

To simulate the set of spam emails that an adversary generates to send to the user to try to bypass their spam filter, I randomly select a set of different spam emails from the corpus, and combine the headers in these email with the headers used in the attack emails described above. The header fields that were modified were the same as those in the *focused attack*, as described in Table 1.

# Chapter 5

# Defenses

## 5.1 Defense Strategies

The adversary may at any time send emails similar to the attacks listed above to maliciously alter the model learned by SpamBayes. It is important to look at defense strategies that mitigate or completely reduce the effect these maliciously designed emails have on the learning process. For exploratory-based attacks, the classification algorithm is often being attacked and for causative-based attacks, the learning algorithm is under attack. However, this does not necessarily mean that in order to defend against exploratory or causative based attacks, we should look at only improving the classification or learning algorithm respectively. Often these two components are closely tied together as it is with SpamBayes. It is useful to create more robust algorithms or additional heuristics for both the learning and classification process to defend against either type of attack.

The attacks in this thesis are all causative in nature, but the *Reject On Negative Impact (RONI) defense* and *dynamic threshold defenses* presented here attempt to make SpamBayes more robust through additional heuristics rather than modifying the learning and classification algorithms. Emails gathered for the training process are also used to approximate what future ham and spam emails will look like. This is not unreasonable if the training set is modified or updated weekly or monthly as ham and spam emails do not significantly change that quickly.

In the *RONI defense*, the trained model is altered so that attack emails are filtered out of the training process. Figure 7 illustrates the process used to filter these attack emails. The idea is to determine whether an email has a negative impact when trained upon, and if so, remove this email from the learning process. The emails gathered for the training set are partitioned into mutually exclusive sets that will be used to both train and test every new email to estimate the impact it will have on future emails.

With the *dynamic threshold defense*, the thresholds used by SpamBayes are changed dynamically according to the scores of emails as shown in Figures 8 and 9. Instead of using fixed thresholds to determine whether an email is classified as *ham*, *unsure*, or *spam*, a heuristic function chooses dynamic thresholds that maximize the true positives and true negatives, while minimizing the false positive and false negatives. Attack emails that shift all or some scores of emails will have a reduced effect since this defense will shift the thresholds accordingly. Since there is a large range that is classified as unsure under the fixed thresholds, this defense will be effective even for large shifts in email scores.

## 5.2 RONI

### 5.2.1 Overview

The *RONI defense* attempts to detect emails that may cause harm if trained on. It does not specifically seek to block any particular attack email, but rather will detect any email, whether it is maliciously designed to alter the trained model or not, that has a harmful effect on the learning process. Essentially, the purpose of this defense is to remove emails from the training set if they will have a harmful affect on classifying future emails as *ham* or *spam*.

To determine the effect on future emails, parts of the training data itself can be used to test the effects a particular email has on the classification of emails in general. To determine the effect on a single email, one would like to train on all the subsets of the training data with and without the email and test on the remaining portions of the training data to accurately determine the effect of the query email. This should be done for every email in the training set. To determine if the query email affects learning, the *RONI defense* compares the accuracy of email classification for each of the subsets with and without the email.

There are several statistics one can use to determine the impact an email has on classification when it is trained on. The difference in false positive and false negative rates before and after training on a particular email is an important statistic to consider. Also, since there is an *unsure* label in SpamBayes, it is important to also look at the change in true positive and true negative rates, since these are not simply the inverses of the false positive and false negative rates respectively. However, this information does not give the full picture of the positive or negative effect of the given email since test emails may cause small changes in score that do not change the actual classification as *spam*, *ham*, or *unsure*. Thus, it is necessary to look at the raw SpamBayes score for each email in the test set. Since I am looking at the negative effect of email classification, two other statistics I look at to determine the impact of a given email are:

$$I_h = \sum_{testhamemails} \max\left(0, I' - I\right) \tag{5.1}$$

$$I_s = \sum_{testspamemails} \min\left(0, I' - I\right) \tag{5.2}$$

where $I$‘ and $I$ are the scores after and before the test email is added to the training set respectively. $I_h$ represents the sum of ham email scores that have shifted towards spam and $I_s$ represents the sum of spam email scores that have shifted towards ham. The total negative impact for a particular email is the sum of the $I_h$ and $I_s$ values.

### 5.2.2 RONI Implementation Methods

For the *RONI defense*, a complete analysis would look at each of the subsets for training and testing because the training data itself may have other attack emails that may influence the effect the query email has on the test emails. For example, an adversary could insert malicious emails that only have a small incremental effect on the training set when other similar emails are present. However, if these emails are not present, the effect of such emails will stand out among the rest and can be detected by this type of defense.

However, time constraints make testing every possible subset for every email in the training set infeasible as this would take far too long to complete. As long as the training and test sets that are chosen from the overall training set are *clean*[1] (no attack emails), it is only necessary to pick one such set to train and test data on. Then, for every email, the defense checks the results of classification after adding the email to the trained model. If this email has a ”significant” negative impact on the results, the defense removes the email from the training set.

In an online training setting, this testing is much easier to perform, as the defense can assume it starts with *clean* data, which can be split up into training and

[1]By the term *clean*, I do not mean free from spam email, but rather free from email that may adversely affect the spam filter when used in the training process

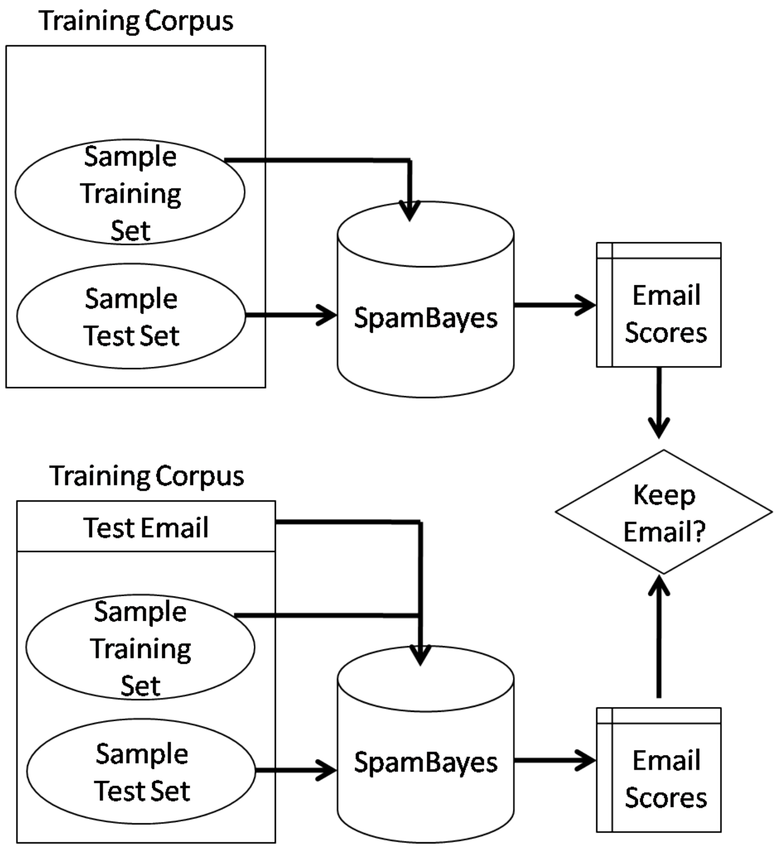

Figure 7: In the *RONI defense*, for every email that is tested, the defense selects a sample train and test set from the training data. The defense trains SpamBayes, with and without the test email, and compares the classification scores to determine if the given test email has a large negative impact. If so, it is discarded from the actual training set. This process is repeated for every email that needs to be tested.

test sets. Next, for every new email that comes in, I can apply the RONI test to check the results of classification on the test set when adding the new email. If this email significantly effects the training results, the defense can simply classify the newly arrived email, but not train on it. To achieve an initial *clean* dataset, it is feasible for a human to give a small (100 or less messages) training set that is hand-picked to ensure that the emails are standard and non-suspicious.

However, this does not work well if there is a large initial set of data that one would like to use for training a spam filter. I do not assume an online training model, and I thus need to either find a *clean* training set from the large initial pool of data, which may contain attack emails ,or mitigate the impact of attack emails from the subset used in training. It is difficult to ensure that the emails chosen for training and testing in the *RONI defense* are not malicious without having a user sort through each email, so I propose a method that attempts to mitigate the impact of attack emails within the training and test sets.

One method of achieving *clean* sets of data for the *RONI defense* is to use a two-pass system that can be extended iteratively. For every query email in the dataset, the defense trains on a random subset that does not contain the query email along with a random test set. If the query email has even a small negative impact, it is marked and not used for future training and test sets. This procedure is for every query email, and on the second pass-through, the training and test sets should have relatively clean data since potentially malicious emails were caught earlier. On the second-pass, the defense can be less selective about marking emails as potentially malicious as it should have a *cleaner* set of data. This process can be repeated iteratively several times until one is confident all the attack emails are found.

Another method, although less ideal, uses only a single-pass to determine the attack emails in the dataset. To approximate a clean training set on the first pass through, the defense uses a very small training set and a larger test set for each query email. Using a small training set limits the number of attack emails that can alter the effects when testing the query email. However, this will not work well when the dataset consists of a large proportion of attack emails.

Orthogonal to the methods proposed above, one can use clustering algorithms to identify similar emails. Only emails at the center of each cluster would need to be used as query emails. This can significantly speed up the *RONI defense*, since only a select few candidate emails would be used as query emails instead of each email belonging to the large initial dataset. Similarity amongst emails can be computed based on the dot product of tokens that emails share in common with one another. Principle component analysis (PCA) can be used to identify emails that are similar to each other after projection onto a small number of dimensions.

## 5.3 Dynamic Thresholding

Distribution-based attacks increase the *spam* score of ham email, but they also tend to increase the *spam* score of spam. Although the *focused attack*, only attempts to target a specific email, it still has some lasting impact on normal email messages by shifting their scores slightly towards *spam*, since other normal messages may share similar features or words as the target email. The *pseudospam attack* will likely have

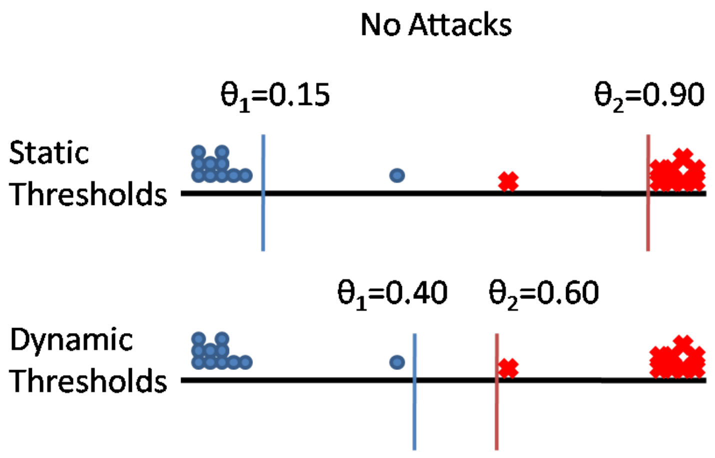

Figure 8: With no attack emails, the standard fixed thresholds performs reasonably well in distinguishing ham and spam emails. Dynamic Thresholds, if there is a clear separation between ham and spam emails, should still be able to distinguish between emails with smaller score differences.

similar tokens in header fields as other spam messages, shifting the score of generic spam email towards *ham*.

Thus with new $\theta_0, \theta_1$ thresholds (Equation 3.5), it may still be possible to accurately distinguish between these kinds of messages after an attack. Based on this hypothesis, I propose a *dynamic threshold defense*, which dynamically adjusts $\theta_0, \theta_1$. With an adaptive threshold scheme, attacks that simply shift all scores will not be effective since rankings are invariant to such shifts. However, more complicated attacks that shift and mix the scores of ham and spam emails so they are closer together are more effective against this type of defense. Thus, this defense is most useful with more naive attacks, and its main purpose is to choose thresholds in a more principled manner than the current SpamBayes implementation.

SpamBayes threshold values, $\theta_0$ and $\theta_1$, are arbitrarily fixed based on normal ham and spam email scores that a user may see in his inbox. This defense attempts to choose thresholds more intelligently by looking at the actual set of email classification scores for a trained model. As it does not attempt to actually remove the attack emails from the training set, this defense can be used in conjunction with the *RONI defense*.

To determine the dynamic values of $\theta_0$ and $\theta_1$, I split the full training set in half. One half is used to train a SpamBayes filter $F$, and the other half is used as a validation set $V$. The training set is split into halves in order to maximize the amount of data SpamBayes can learn on, while also having an ample testing set to accurately

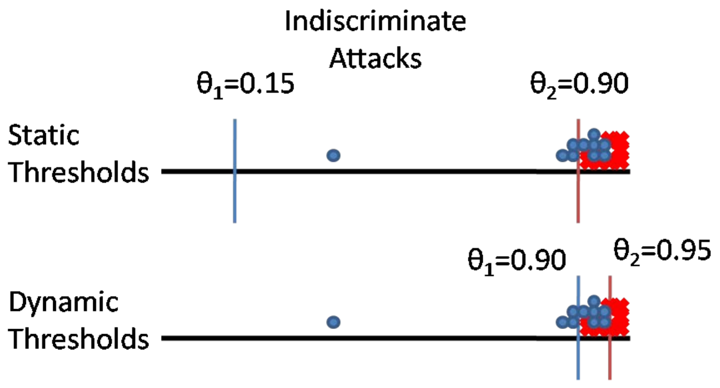

Figure 9: In the presence of an attack that shifts all scores towards spam, dynamic thresholds are still able to distinguish between some ham and spam emails provided there is some small separation between the two classes of emails.

determine the distribution of ham and spam email scores. It can be argued that for large training sets, it may be more effective to train on most of the emails leaving only a few thousand emails into the validation set $V$.

Using $F$, the defense obtains a score for each email in $V$. From this information, it is possible to pick threshold values that more accurately separate future ham and spam emails. I define a utility function for choosing threshold $t$:

$$g(t) = N_{S,<}(t)\left(N_{S,<}(t) + N_{H,>}(t)\right)^{-1} \tag{5.3}$$

where $N_{S,<}$ is the number of spam messages with scores less than $t$, and $N_{H,>}$ is the number of ham messages with scores greater than $t$. I select $\theta_0$ so that $g(\theta_0)$ is 0.05 or 0.10, and I select $\theta_1$ so that $g(\theta_1)$ is 0.95 or 0.90, respectively. These are chosen based on experimental evidence, as these threshold values tend to give the best results in classifying ham and spam email.

# Chapter 6

# Experiments

## 6.1 Experimental Framework

In this section, I show the experimental results from the attacks and defenses presented in Chapters 4 and 5. Both of the attacks are highly successful in subverting the SpamBayes spam filter while only controlling a small portion (10% or less) of the training set. The defenses presented show promise against some variations of the attacks, while having difficulty in preventing misclassifications by SpamBayes on specific targeted emails that do not have much in common with other email the filter normally sees.

For my experiments, I use the Text Retrieval Conference (TREC) 2005 spam conference corpus [3], which is based on the Enron dataset [11]. This corpus contains 92,189 emails of which 52,790 are spam and 39,399 are ham emails. I use this corpus as it has a large number of emails, comes from a real-world source, and the added spam does not have obvious artifacts that can make it easily differentiable from ham. Also, I use a large USENET corpus in some of my attacks, which is used as a distribution of words that may be commonly found in emails.

To validate my results, I use $K$-fold cross-validation with $K = 10$. Cross-validation is a technique used to prevent against type III errors, or errors when testing hypothesis suggested by the data. This technique involves partitioning the training set into $K$ subsets. $K - 1$ subsets are used for the training process and the lone remaining subset is used as the test set. This is repeated $K$ times such that each subset is used in both the training and testing process. These $K$ repetitions are then averaged to produce a single estimation of the correct result.

The experiments are run on the DETER [4] Testbed. I use several machines or nodes from the testbed, ranging from 2GHZ single core to 3.0GHZ dual core machines,

| **Parameter** | **Focused Attack** | **PseudoSpam Attack** | **RONI Defense** | **Threshold Defense** |
|---|---|---|---|---|
| **Training set size** | 2,000, 10,000, | 2,000, 10,000 | 2,000, 10,000 | 2,000, 10,000 |
| **Test set size** | 200, 1,000 | 200, 1,000 | N/A | 200, 1,000 |
| **Spam prevalence** | 0.50, 0.75, 0.90 | 0.50, 0.75, 0.90 | 0.50 | 0.50, 0.75, 0.90 |
| **Attack fraction** | 0.001,0.005,0.01, 0.02,0.05,0.10 | 0.001,0.005,0.01, 0.02,0.05,0.10 | 0.10 | 0.001,0.005,0.01, 0.02,0.05,0.10 |
| **Folds of validation** | 10 | 10 | N/A | 10 |
| **Target Emails** | 20 | N/A | N/A | N/A |

Table 2: Parameters used in the experiments

to run each of my experiments across a wide range of parameters as shown in Table 2. Most of the attacks and defenses were run using a 2,000 and 10,000 email training set and I present the results when using the 10,000 email training sets in my experiments. The size of the training set and percentage of spam in the training set have a negligible impact on the results.

## 6.2 Attacks

### 6.2.1 Focused

For the *focused attack*, I setup the experiment by first randomly selecting a set of target emails from the TREC corpus. The target emails chosen are ensured to contain at least 100 words in the message body. I then create the set of training data for SpamBayes by randomly selecting a large corpus of ham and spam email along with a set of attack spam email, which is generated by the method illustrated in Section 4.3. Finally, I collect the results across each of the 10 folds of cross-validation. Based on my results, the *focused attack* did not appear to have any significant impact on normal ham and spam emails.

In Figure 10, I examine the impact of the *focused attack*, while varying the knowledge of the attacker. In this figure, the adversary has control of 10% of the training set for each of the attack variations and the initial inbox size contains 10,000 emails. All of the targeted emails were initially classified as ham before any of the attack emails were added. When the adversary only has knowledge of approximately 10% of the targeted email's tokens, the attack is very rarely successful in misclassifying the email as *spam*, but over half the time the email is misclassified as *unsure*. However, when the adversary is able to guess only 30% of the email, every email is misclassified

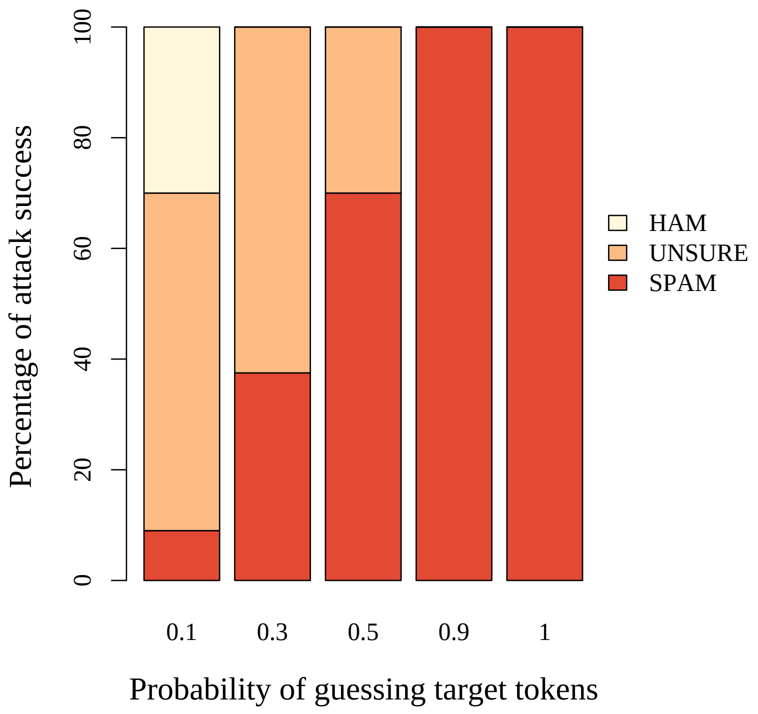

Figure 10: Effect of the *focused attack* as a function of the percentage of target tokens known by the attacker. Each bar depicts the fraction of target emails classified as *spam*, *ham*, and *unsure* after the attack. The initial inbox contains 10,000 emails (50% spam).

by SpamBayes. If the attacker knows a significant portion of the email (90%), I show that all the time the email is classified as *spam*. Thus, even when the adversary only has modest knowledge of the email's contents, many targeted emails will no longer be classified as *ham*.

Next, I examine the effect of increasing the number of attack emails as shown in Figure 11. In this attack, the initial inbox contains 10,000 emails, 50% of which are spam, and the attacker is able to guess 50% of the email's content. The $x$-axis is the number of messages in the attack and the $y$-axis is the percent of messages misclassified. From this, it is clear that the adversary only needs to poison a very minimal number of emails, in relation to the size of the training set, for the attack to be effective. I suspect that this is due to the distribution of words in emails, as word selection in the English language follows a Zipf distribution [25]. Thus, many words in the attack email are not seen in other email, even with a large training set, so the words in the targeted email become more indicative of spam with only a few attack emails.

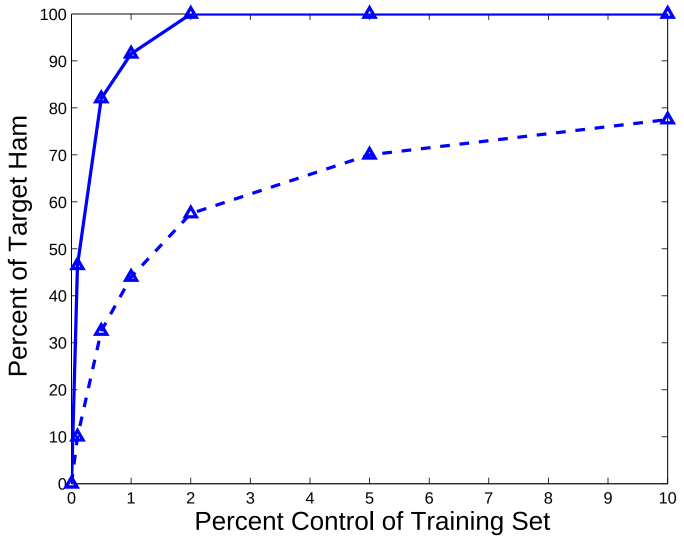

Figure 11: Effect of the *focused attack* as a function of the number of attack emails with a fixed percentage ($p$=0.5) of tokens guessed by the attacker. The dashed line shows the percentage of target ham messages classified as *spam* after the attack, and the solid line the percentage of targets that are *unsure* after the attack. The initial inbox contains 10,000 emails (50% spam).

Looking at the token distribution in the targeted emails before and after the attack provides further insight into the success of the attack. In Figure 12, the three graphs represent three targeted emails: (from left to right) ham misclassified as spam, ham misclassified as unsure, and ham classified correctly as *ham*. The blue $\bigcirc$'s are tokens that were not guessed correctly by the attacker and the red $x$'s are tokens chosen correctly by the attacker. The line $y = x$ separates tokens that increase due to the attack (above the line) and tokens that decrease due to the attack (below the line).

These graphs indicate that tokens chosen by the attacker have a significant increase in spam score as indicated by a large number of red $x$'s in the upper right corner, while tokens not chosen only decrease slightly as these tokens remain closer to the $y = x$ line. Since the increase in score is more significant for included tokens than the decrease in score for excluded tokens, the attack has substantial impact even when the attacker has a low probability of guessing tokens as seen in Figure 10. Further, the before/after histograms in Figure 12 provide a direct indicator of the attack's success.

If the adversary has no knowledge of the email, it is best for the attacker to include words that may come from any email. In Figure 13, I look at the success of

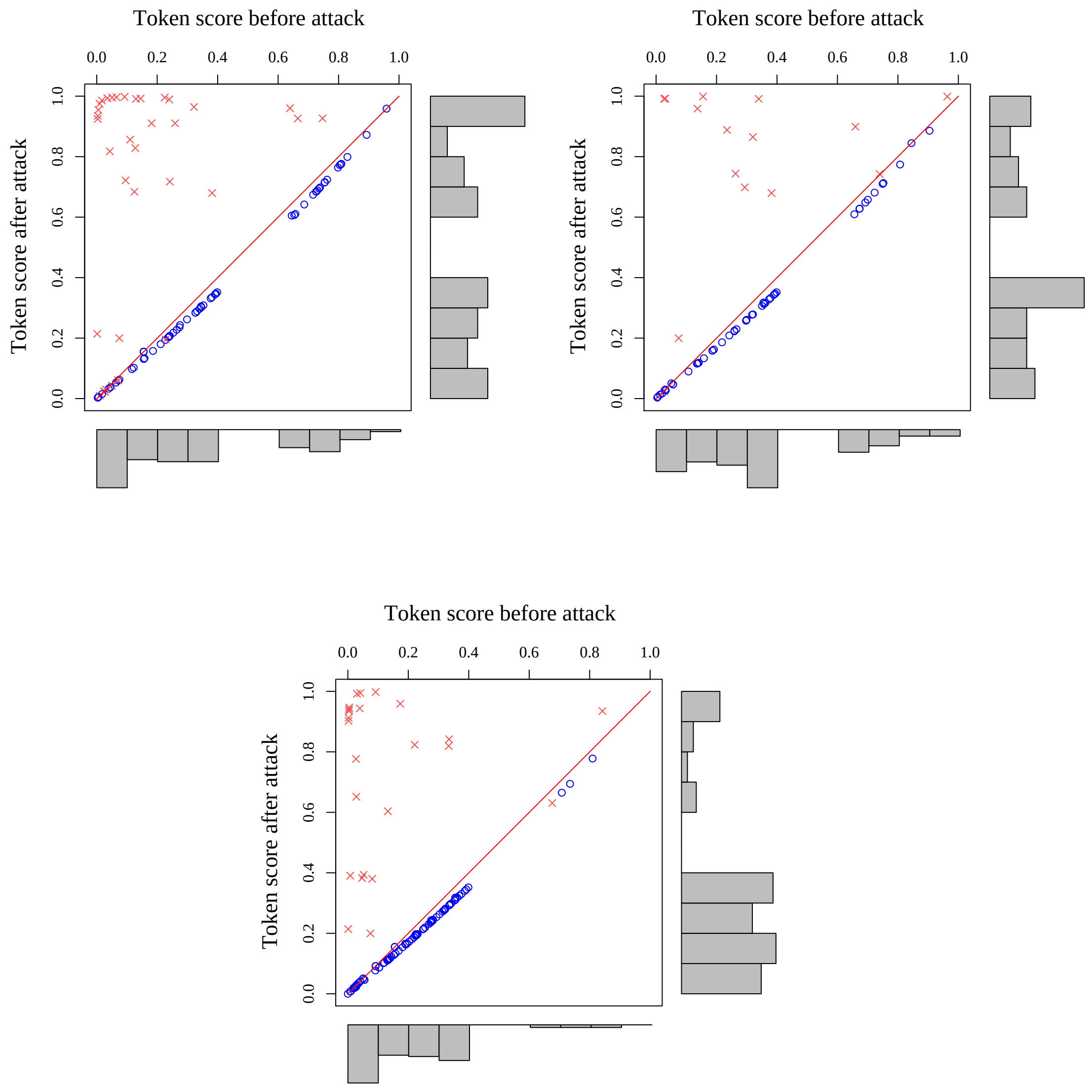

Figure 12: Effect of the *focused attack* on three representative emails— one graph for each target email. Each point is a token in the email. The $x$-axis is the token spam score in Equation (3.2) before the attack (0 means ham and 1 means spam). The $y$-axis is the spam score after the attack. The red $x$'s are tokens that were included in the attack and the blue $\bigcirc$'s are tokens that were not in the attack. The histograms show the distribution of spam scores before the attack (at bottom) and after the attack (at right). Only ten percent of the tokens are successfully chosen by the adversary.

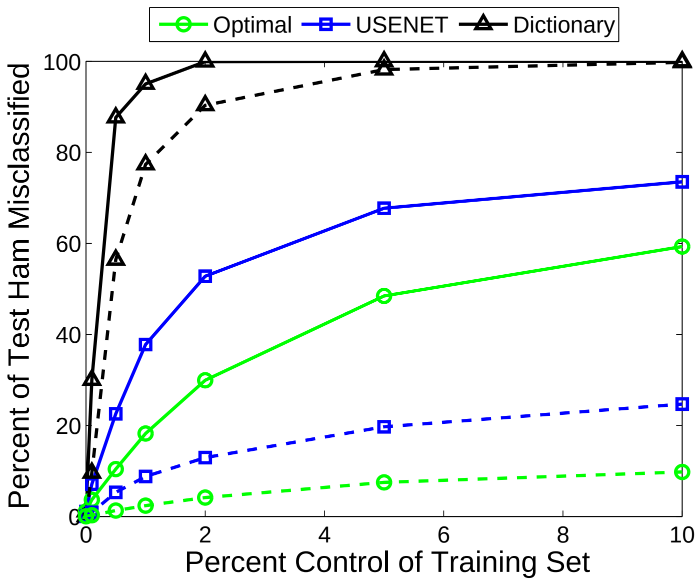

Figure 13: Three attacks on initial training set of 10,000 messages (50% spam). I plot percent of ham classified as *spam* (dashed lines) and as *spam* or *unsure* (solid lines) against the attack as percent of the training set. I show the optimal attack (black △), the USENET dictionary (blue □), and the Aspell dictionary (green ○). Each attack renders the filter unusable with as little as 1% control (101 messages).

the attack on general ham email when the attacker chooses words that are optimal (all words from the set of ham test email), the most commonly used 90,000 words from the USENET corpus, or the 90,000 word Aspell English dictionary. The figure shows that with 10% control of the training set, each of the attack variants cause over 50% misclassification of ham email. Also, with as little as 101 email messages, the adversary can cause enough misclassifications to frustrate the user into turning off the spam filter.

Although only a few messages are needed to corrupt the filter, the number of tokens in each email are quite large since this attack attempts to cover all or many of the possible tokens in any given ham email. Preliminary experiments show that reducing the number of words from 90,000 to 10,000 words from the USENET corpus gives the same results as the using the dictionary.

### 6.2.2 Pseudospam

I setup the experiment for the *pseudospam attack* by first randomly selecting a spam header to be used as the base header for the attack. I then create the set of

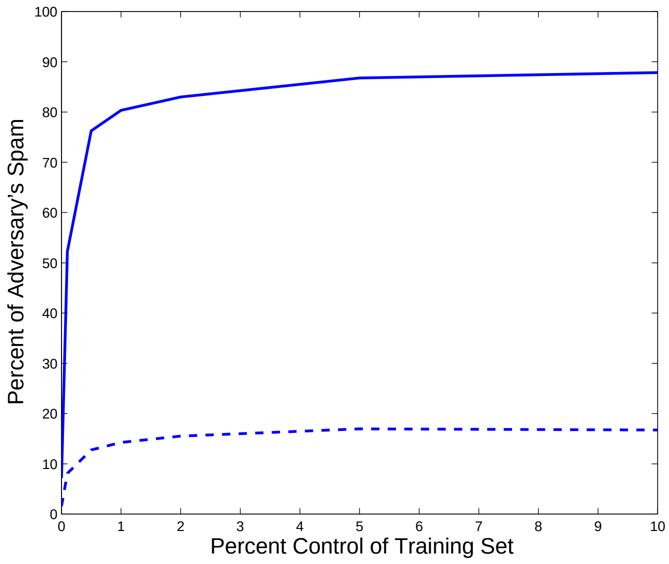

Figure 14: Effect of the *pseudospam attack* as a function of the number of attack emails. The dashed line shows the percentage of the adversary's messages classified as *ham* after the attack, and the solid line the percentage that are *unsure* after the attack. The initial inbox contains 10,000 emails (50% spam).

attack emails that look similar to ham emails (see Section 4.4). Next, I randomly select a mutually exclusive set of training data from the TREC corpus. After SpamBayes trains on the data along with the attack emails, I test the classifications against both a set of spam email generated by the adversary along with normal ham and spam email.

Figure 14 demonstrates the effectiveness of the *pseudospam attack* as the adversary gains control over a growing portion of the training set. The y-axis shows the percentage of test spam email that is classified as *ham*, *spam*, or *unsure*. The solid blue line is the percent of spam classified as *ham* or *unsure* spam while the dashed blue line is the percent of spam classified as *ham*. From initially being able to correctly classify over 90% of these spam emails, SpamBayes correctly labels less than 20% of spam after only a couple hundred attack emails. This value quickly levels off at slightly fewer than 15%. The number of emails classified as *ham* also reaches and stays at 20% relatively quickly. The reason the curve levels off is due to the contents of the spam messages still having tokens with high spam indicators. Only the headers are being targeted in this attack, which is sufficient for a large percentage of spam emails for the adversary to be satisfied in getting some of his spam emails through to the user.

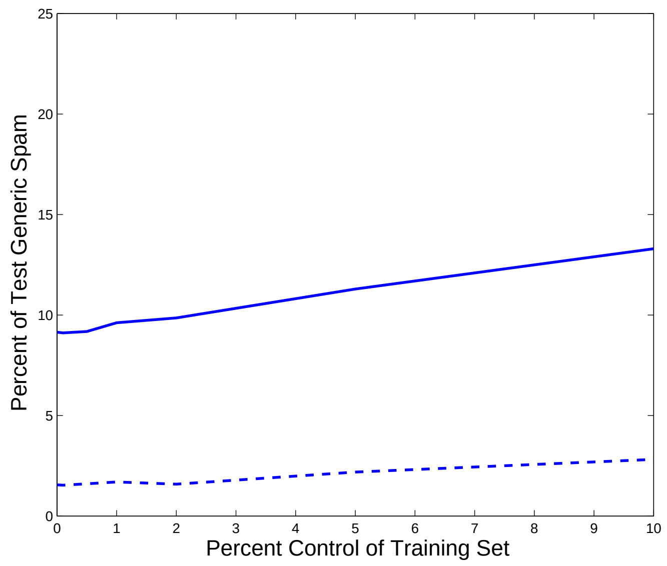

Figure 15: Effect of the *pseudospam attack*, as a function of the number of attack emails. The dashed line shows the percentage of the normal spam messages classified as *ham* after the attack, and the solid line the percentage that are *unsure* after the attack. Surprisingly, training the attack emails as ham causes an increase in misclassification of normal spam messages. The initial inbox contains 10,000 emails (50% spam).

The attack has a minimal effect on normal ham and spam email classifications. Since the headers that the adversary generates are being poisoned in order to be considered *ham* indicators, other spam email messages will still be correctly classified since they would not have the same header fields as the adversary. Also, ham email messages may have lower *spam* scores according to the SpamBayes classifier since these messages may contain similar words as the attack emails generated by the adversary.

In the cases where the *pseudospam attack* emails are actually labeled as *spam* in the training set, the SpamBayes classifier begins to classify generic spam email incorrectly. Figure 15 indicates an increase in spams mislabeled as both *unsure* and *ham* as the number of attack emails increases. Almost 15 percent of spam email is misclassified. This effect is due to the attack emails not containing traditional spam words, which causes normal indicators of spam to become slightly more indicative of ham (see Section 4.3 for a description of why tokens become more indicative of ham when tokens are not present in the training spam emails). This version of the attack does not cause much of an impact on normal ham messages. The specific spam emails from the adver-

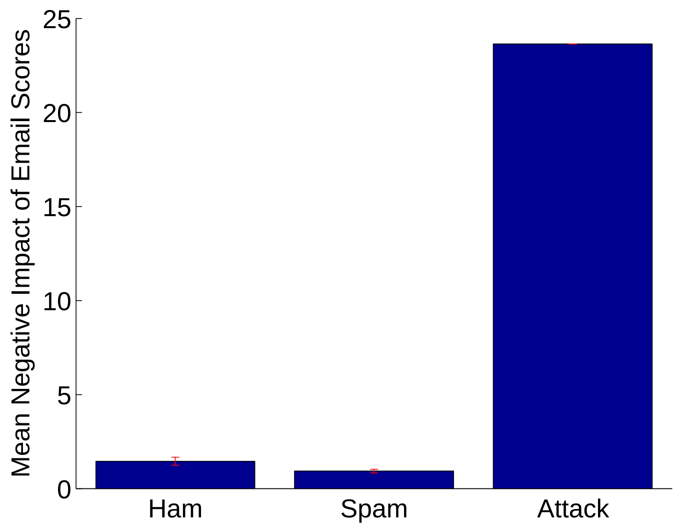

Figure 16: The total negative impact is determined by adding the total increase in ham email score along with the total decrease in spam email score. The standard error is shown along with each bar in the graph. The attack email used in the standard dictionary attack email. A *clean* set of email is used for training and testing. The initial inbox contains 10,000 emails (50% spam).

sary are still misclassified, though at a much lower rate than if the attack emails were trained as ham.

## 6.3 Defenses

### 6.3.1 RONI

In order to determine the effectiveness of the *RONI defense*, the RONI query set, which is used to determine the negative (or positive) impact of adding a particular email, consists of a sample of email from the entire 10,000 email training set. One hundred ham and spam emails are randomly sampled into the query set along with one of the attack emails. Since the attack emails are very similar, only one of them is placed in this query set. The sampled training set contains 20 emails and the test set contains 100 emails, each of which is randomly chosen from the 10,000 email training set. This procedure is repeated ten times, which is averaged for the results shown in the figures.

Figure 16 uses a sample training set of size 20 that does not contain any attack emails. The attack email used is the standard dictionary email presented earlier. Most

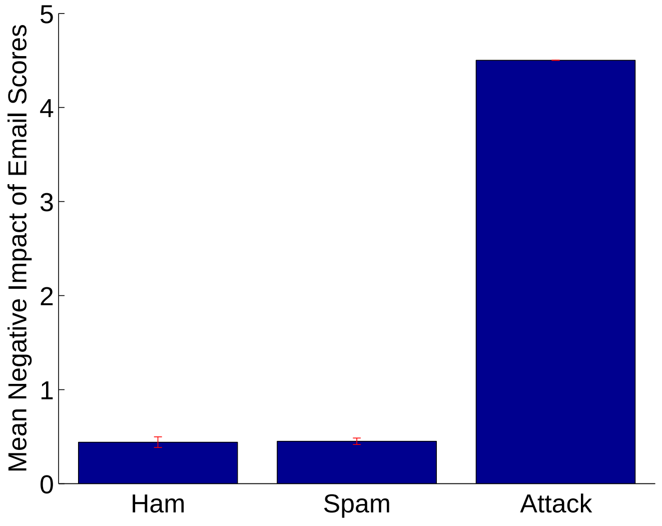

Figure 17: The total negative impact is determined by adding the total increase in ham email score along with the total decrease in spam email score. The standard error is shown along with each bar in the graph. The attack email used in the standard dictionary attack email. A random sample of attack and non attack emails are used in the training and test set. The initial inbox contains 10,000 emails (50% spam).

email trained as spam only shift ham email scores more towards *spam*, and do not negatively impact spam email classification while the reverse is true for email trained as *ham*. The positive impact of emails are not measured in this graph nor is it taken into account by RONI. It is clear that the dictionary attack email has a large negative impact that far exceeds any negative impact a standard email would have. In this situation, there are many threshold values that can be chosen that would throw away the dictionary email, and not train on this email in the training set.

Figure 17 more realistically uses a sample training set of size 20 that may contain attack emails, as the training set is sampled from all emails in the 22,222 email training set (of which 2,222 are attack emails). The attack email here still has the largest negative impact over all other emails used in the query set, but the difference is not as large. The *RONI defense* would still throw away the dictionary emails, since it has the largest negative impact on classification, but certain heuristic thresholds may also cause some normal email to be thrown away in the training set. Also, since the attack emails are also present in the training set, the total negative impact for all email decreases by over a factor of two, since the initial training set used in the RONI test is not free from emails that may have a large negative effect on email classification.

The *RONI defense* does not do well at all in picking out emails from the *focused attack* or the *pseudospam attack*. Attack emails used for these attacks do not indiscriminately affect email that is already present in the training set. In fact, the attack emails used in these attacks affect only future email, and the *RONI defense* only uses presently known email to determine the negative impact of training on a particular email. Thus, the *RONI defense* is a heuristic method, which is effective against *indiscriminate* attacks that undermine the *availability* or *integrity* of the spam filter. *Targeted* based attacks will slip through this defense unless such attacks also have a negative impact on normal email as well.

### 6.3.2 Thresholding

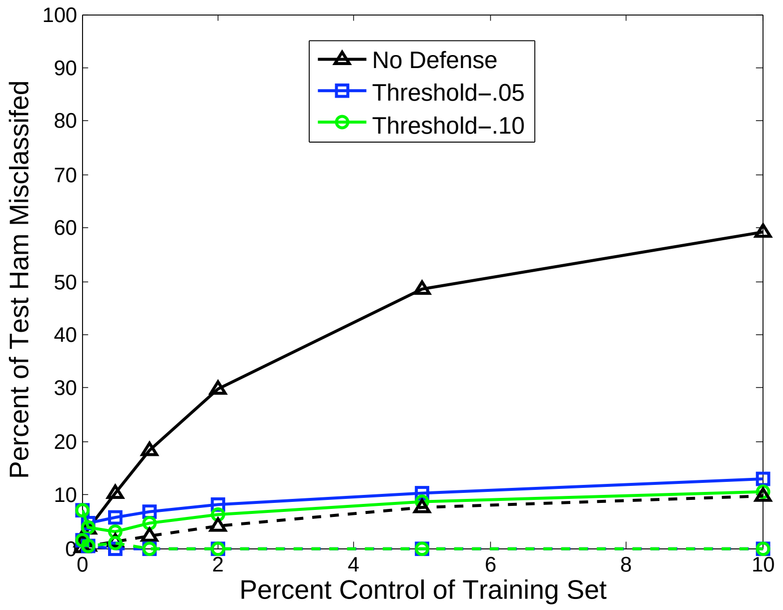

Figure 18: Effect of the *threshold defense* on the classification of ham messages with the dictionary based attacks. I use a 10,000 message inbox training set of which 50% are spam. The solid lines represent ham messages classified as *spam* or *unsure* while the dashed lines show the classification rate of ham messages as *spam*. Threshold-.05 has a wider range for *unsure* messages than the Threshold-.10 variation.

To compare the results of the attacks against the *dynamic threshold defense*, we run the attacks as presented before against standard SpamBayes along with the *threshold defense*. The *threshold defense* is implemented on top of the SpamBayes training and classification functions, adding the changes after calls are made into the filter.

First, I examine the effect of choosing different thresholds against dictionary based attacks. Figure 18 shows the results of the dictionary attack against SpamBayes and two variations of the threshold defense. From this, it is clear that none of the ham emails are classified as *spam* and only a modest amount, 10%, are classified as *unsure*. When there are no attack emails, no ham messages are classified as *spam*, but there are a few still classified as *unsure*. Experimenting with different heuristic functions for picking the dynamic thresholds may yield better results when no adversarial data is present.

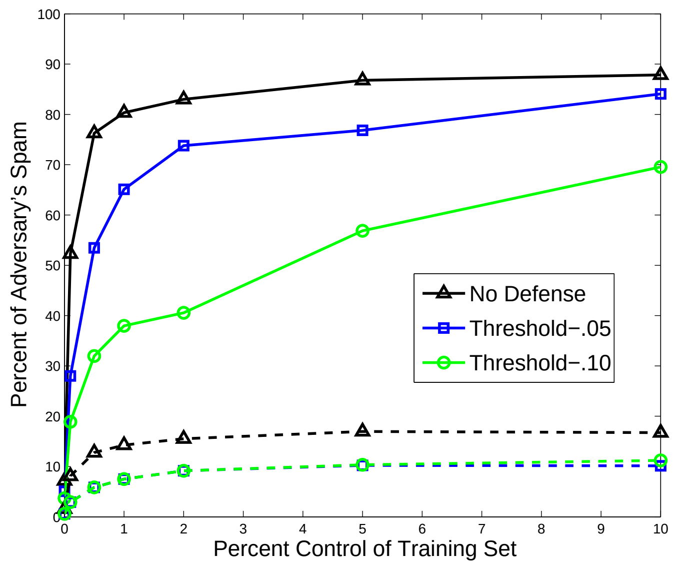

Figure 19: The dynamic thresholds are able to separate a larger portion of the adversary's spam emails. However, a large percentage of these emails are still classified as *unsure*.

However, not shown, are the classification of spam emails, which are all classified as *unsure* as the attacker gains 10% control of the training set. When looking at the raw email scores, this is due to SpamBayes classifying all spam email and 10% of the ham email as being purely spam with a score of 1.0. Although the dynamic thresholds do better than traditional SpamBayes classifications, these thresholds are not successfully able to separate out all ham and spam email from each other. Since the SpamBayes classifier can only give a max spam score of 1.0, many emails have no separation in the email score. It is likely, that when combined with other defensive measures, the dynamic thresholds will be more accurate in email classification since there will be a some separation between the ham and spam emails.

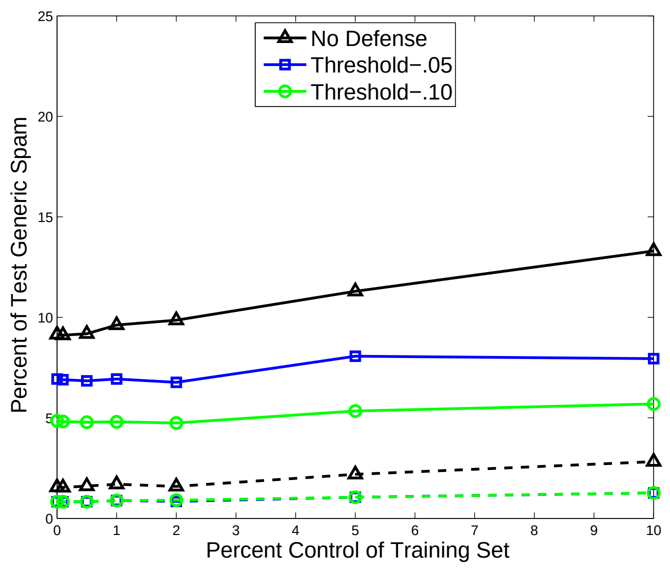

Figure 20: The dynamic thresholds reduce the number of misclassifications. As the attack increases, the number of misclassifications stay fairly flat instead of increasing.

For the more specific variations of the *focused attack*, the threshold defense has no impact on the misclassification rate as the same number of emails are still incorrectly classified by SpamBayes. Since the *focused attack* does not have an impact on normal email, the threshold defense is not able to shift any of the thresholds in a meaningful way to capture the effect of a future unknown email from being misclassified.

Figures 19 and 20 shows that the dynamic thresholds are able to reduce the negative impact of the *pseudospam attacks*. However, both figures show that this defense is far from perfect against these attacks as still a significant number of spam messages are misclassified as *unsure*. In Figure 19, the number of misclassifications still increase as the adversary gains control over a larger portion of the training set, whereas in Figure 20 the number of misclassifications of spam email stays fairly flat as the attack increases.

Overall, the *dynamic threshold defense* appears to mitigate some of the impact of the attacks. However, it is not strong enough, in its present form, to completely prevent indiscriminate attacks from affecting SpamBayes' email classification. Attacks that are more targeted in nature, especially those that do not share many features with other emails found in the training set, are extremely difficult to prevent using this defense since it has no way of altering the thresholds based on the current data. This defense is best used orthogonally with additional defenses that are more robust to attacks on the learning component of SpamBayes, as changing the thresholds dynamically only attempts to reduce the harm of an already compromised learner model.

# Chapter 7

# Conclusion

## 7.1 Discussion

Using the framework established in Section 4.1, I have demonstrated the effectiveness of causative attacks against SpamBayes that violated both the availability and integrity of the spam filter. The adversary was given realistic control over the training process of SpamBayes, by limiting the header fields the attacker can modify and the amount of control over the training set. Also, the labels over the attack emails are controlled by the user, and as such, can only be labeled as *ham* if the email looks like a legitimate message while other messages are labeled as *spam*.

I show that the attacks are very successful against the learning algorithm used by SpamBayes. The *focused attack* is able to cause the learner to classify many specific messages as *unsure* when the adversary has little knowledge of the message and as *spam* when the adversary has quite a bit of information regarding the message. A variation of the *focused attack*, the dictionary attack, is able to cause over half of all ham messages to be misclassified. In the *pseudospam attack*, the adversary is able to get 90% of his spam messages labeled as either *unsure* or *ham*. In the unsuccessful case where the attack emails are instead trained as *spam*, generic spam messages are increasingly successful in reaching the user's inbox.

My defenses show initial promise in being able to defend against the more indiscriminate attacks. Although these defense mechanisms are far from perfect, the *dynamic threshold defense* is able to reduce the impact of the distribution and *pseudospam attacks*. Also, the defenses can be implemented in combination with one another, improving the SpamBayes classifier further. Another benefit to these defenses is that the underlying algorithm is unchanged, so improvements made to the learning or classifying component of the filter would not hinder the defense strategies presented here.

Rejecting emails based on their negative impact can be applied to other machine learning algorithms. It should be possible to detect the effect of training on incrementally new instances of data, in other classifiers other than spam filtering. Similarly, the *dynamic threshold defense* can be applied to any other algorithm that uses thresholds to determine the classification of the underlying data. These defense strategies can be combined with each other as well as more defenses that change the underlying machine learning algorithm.

An attacker can still circumvent around these attack strategies. If an adversary's goal is to change the classification of future messages only, the current defenses can do little to circumvent this problem as shown in the experimental results. Also, by using a large number of messages that very slowly change the email scores, the *RONI defense* may not be as effective in detecting the negative impacts of the malicious messages. Ideally, the *RONI defense* would be able to holistically determine the effect of adding larger sets of emails that have similar tokens that, when trained on, have a significant impact on email classification. A cluster based approach for tokens in the *RONI defense* may be able to determine which tokens are malicious. Another improvement to this defense is to reject only the portions of each individual email that has the spurious features that can cause email misclassifications. Training on the remaining information present in the email, will still be helpful for the spam filter.

The *dynamic threshold defense* is only able to detect shifts in email scores from changing the classifications of specific messages or only altering the overall email scores of only ham or spam messages. As shown in my results, dynamically altering the thresholds is useful in reducing the impact of many attacks. Due to the nature of this strategy, this mechanism will not be able to completely reduce an adversary's ability to cause havoc through either availability or integrity attack strategies since the model learned by the spam filter stays the same. Dynamic thresholds should always be used in conjunction with other defenses as it is a more intelligent way to heuristically choose cutoffs for labelling data.

## 7.2 Future Work: Next Steps

There are many directions that I would like to take this work in the future. Attack and defense strategies can be improved and extended into areas other than spam filtering. Listed below is a list of areas I would like to explore in further research.

- The attacks presented here can be extended into a single general attack that is effective when trained as either ham or spam. The *pseudospam attack* is effective as it focuses on attack the headers whereas the *focused attack* focuses on the message bodies. Combining these attacks could yield attack messages that subvert the effectiveness of classifying both generic ham and spam email when trained as *spam* while also allowing the adversary to craft specific spam messages to bypass the filter if the attack messages are trained as *ham*.

- My attacks can also be further extended to circumvent the *RONI* and *dynamic threshold defenses* completely. Also, I would like to extend my current defenses and develop new defenses to more completely defend against the current and future variations of the attacks.

- Combine the defenses presented here to determine their effectiveness when used together. Also, create defenses that can be orthogonally implemented alongside the current ones that improve the learning process of SpamBayes.

- Expand the *RONI defense* to work at a more defined feature level by clustering tokens. The tests used in this defense can be used on the specific tokens in emails or clusters of tokens to determine which tokens have a negative impact on classifying email.

- Similarly, I also plan to examine the vulnerability of other classes of security-sensitive learning systems, such as intrusion detection systems, and worm and virus detection systems. I plan to extend the attacks presented here to these systems by working along the attack framework presented by in Section 4.1.

- The defenses implemented here should be usable within other machine learning systems; in particular, systems that use statistical machine learning algorithms to both train and classify data. Systems that use a training and classifying processes would be able to use variations of the *RONI* and *dynamic threshold defenses* presented in this thesis. Extending these defenses would give us a better understanding of their viability in other machine learning systems.

# Acknowledgments

The work presented here in this thesis was also worked on by others in the Security in Machine Learning Research group at UC Berkeley. I would like to thank all of those in this group, which includes Marco Barreno, Fuching Jack Chi, Dr. Anthony D. Joseph, Blaine Nelson, Dr. Satish Rao, Benjamin I. P. Rubinstein, Charles Sutton, Dr. J. D. Tygar, and Kai Xia. I would also like to give special thanks to my advisor, Dr. Anthony D. Joseph, for helping me all along in this project and guiding me in writing this report. I give my thanks to Dr. Doug Tygar for reading and revising this report.

This work was supported in part by the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148 and #07-012) and Siemens; and in part by the cyber-DEfense Technology Experimental Research laboratory (DETERlab), which receives support from the Department of Homeland Security Homeland Security Advanced Research Projects Agency (HSARPA award #022412) and AFOSR (#FA9550-07-1-0501). The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any funding agency, the State of California, or the U.S. government.

# Bibliography

[1] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the ACM Symposium on InformAtion, Computer, and Communications Security (ASIACCS'06)*, March 2006.

[2] Simon P. Chung and Aloysius K. Mok. Allergy attack against automatic signature generation. In *Recent Advances in Intrusion Detection (RAID)*, pages 61–80, 2006.

[3] Gordon Cormack and Thomas Lynam. Spam corpus creation for TREC. In *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, July 2005.

[4] DETER. Deter network security testbed. `http://http://www.deterlab.net/`.

[5] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Salvatore J. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Proceedings of the Workshop on Data Mining for Security Applications*. Kluwer, 2002.

[6] Ronald A. Fisher. Question 14: Combining independent tests of significance. *American Statistician*, 2(5):30–30J, 1948.

[7] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 59–68, 2006.

[8] GNU. Aspell. `http://aspell.net/`.

[9] Paul Graham. A plan for spam. `http://www.paulgraham.com/spam.html`, August 2002.

[10] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[11] Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, July 2004.

[12] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining*, May 2003.

[13] Yihua Liao and V. Rao Vemuri. Using text categorization techniques for intrusion detection. In *Proceedings of the 11th USENIX Security Symposium*, pages 51–59, August 2002.

[14] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005.

[15] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

[16] Tony Meyer and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, July 2004.

[17] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'02)*, pages 1702–1707, 2002.

[18] James Newsome, Brad Karp, , and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, May 2005.

[19] James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, September 2006.

[20] Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, March 2003.

[21] Cyrus Shaoul and Chris Westbury. A USENET corpus (2005-2007), October 2007. `http://www.psych.ualberta.ca/~westburylab/downloads/usenetcorpus.download.html`.

[22] SpamBayes. A Baysian anti-spam classifier written in Python. `http://spambayes.sourceforge.net/`.

[23] S. J. Stolfo, W. J. Li, S. Hershkop, K. Wang, C. W. Hu, and O. Nimeskern. Detecting viral propagations using email behavior profiles. In *ACM Transactions on Internet Technology*, 2004.

[24] Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[25] G. Zipf. *Selective Studies and the Principle of relative Frequency in Language.* Harvard University Press, Cambridge, MA, 1932.